COMPUTATIONAL FLUID DYNAMICS ON A MASSIVELY PARALLEL COMPUTER

Dennis Jespersen, CFD Branch, NASA/Ames
and
Creon Levit, NAS Applied Research Branch, NASA/Ames

Abstract. We have implemented a finite-difference code for the compressible Navier-Stokes equations on the Connection Machine, a massively parallel computer. The code is based on the ARC2D/ARC3D program and uses the implicit factored algorithm of Beam and Warming. The code uses odd-even elimination to solve linear systems. We give timings and computation rates for the code, and compare with a Cray XMP.

COMPUTATIONAL FLUID DYNAMICS ON A MASSIVELY PARALLEL COMPUTER

Dennis Jespersen, CFD Branch NASA/Ames
and
Creon Levit, NAS Applied Research Branch, NASA/Ames

- Goal: implement a real CFD code (ARC2D/ARC3D) on a massively parallel computer, the Connection Machine.
  - > Implicit

- Is Connection Machine useful for CFD?

OUTLINE

- CM architecture
- ARC2D
- *Lisp
- Odd-even elimination
- Current status and results
- Problems and prospects

CM ARCHITECTURE

- Very large number of physical processors (up to 64K)
- SIMD
- Single-bit processors
- 2K words (32 bits each) memory per processor
- Floating point accelerator available
- Context bit (each processor on or off)
- Peak performance (computational kernels) 2–4 Gflops

CONNECTIONS

- 16 processors per chip
- Chips wired in 12-dimensional hypercube
  - > If distance=power of 2, communication in 2 hops
- General router
- Nearest-neighbor (NEWS) available and fast
  - > (Reminiscent of ILLIAC)
- Wrap-around at edges

VIRTUAL PROCESSORS

- Each physical processor can be configured as $1, 2, 4, 8, \ldots$ virtual processors.

VP ratio	1	2	4	8	16	32	64
memory/processor (words)	2K	1K	512	256	128	64	32

- Transparent to user.
- Arithmetic slows down (slightly slower than linear)
- Communication relatively faster
  > Memory references that were off-chip may become on-chip.

ARC2D

- 2D Compressible Navier-Stokes

$$\partial_t Q + \partial_\xi E + \partial_\eta F = \mathrm{Re}^{-1}(\partial_\xi E_v + \partial_\eta F_v)$$

- Central space differencing, implicit Euler in time, factored, delta form:

$$(I + \Delta t \delta_\xi A^n - \Delta t \frac{1}{\mathrm{Re}} \delta_\xi R^n)(I + \Delta t \delta_\eta B^n - \Delta t \frac{1}{\mathrm{Re}} \delta_\eta S^n)\Delta Q^n$$

$$= -\Delta t(\delta_\xi E^n + \delta_\eta F^n - \frac{1}{\mathrm{Re}}(\delta_\xi E_v^n + \delta_\eta F_v^n))$$

- Pulliam-Chaussee diagonal form:

$$T_\xi(I + \Delta t \delta_\xi \Lambda_\xi - \Delta t \frac{1}{\mathrm{Re}} \delta_\xi \Lambda_{\xi,v})$$

$$N(I + \Delta t \delta_\eta \Lambda_\eta - \Delta t \frac{1}{\mathrm{Re}} \delta_\eta \Lambda_{\eta,v}) T_\xi^{-1} \Delta Q^n = \mathrm{rhs}$$

ARC2D on CM

- One grid point per (virtual) processor.
- Write in *Lisp
  > No line-by-line translation
- Shear-layer
- Cylinder
- Typical size 128×64 (8K processors, VP ratio $= 1$)
- CM well-suited for explicit time-step methods (e.g., Runge-Kutta)

IMPLICIT SOLVER: ODD-EVEN ELIMINATION

- Closely related to cyclic reduction
- One row corresponds to one processor
- Communication is always to processors 2^k distant
- $2^n \times 2^n$ system solvable in n stages
- Algorithm makes no sense on uniprocessor machines
- Stability not a problem in practice
  - > Some stability results provable, e.g. $B(-\Delta t, 1, \Delta t)$

ODD-EVEN ELIMINATION: ALGORITHM

$Ax = y$

For $k = 0, 1, \ldots, \log_2 n - 1$,

$$\begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix}$$

$$m = 2^k$$

$$a_i \leftarrow a_i / b_i$$

$$c_i \leftarrow c_i / b_i$$

$$y_i \leftarrow y_i / b_i$$

$$b_i \leftarrow 1 - a_i c_{i-m} - c_i a_{i+m}$$

$$y_i \leftarrow y_i - a_i y_{i-m} - c_i y_{i+m}$$

$$a_i \leftarrow -a_i a_{i-m}$$

$$c_i \leftarrow -c_i c_{i+m}$$

ODD-EVEN ELIMINATION:PERIODIC

- Same code
- Same cost
- Periodic solves no more costly than nonperiodic solves

PROGRESS

- 2D Explicit (Runge-Kutta)
- 2D Implicit (scalar and block tridiagonal)
  - > Shear layer
  - > Cylinder (low Re)
- 3D Explicit (Runge-Kutta)

TIMINGS

- Timings, Mflop rates (16k processors, implicit scalar tridiagonal algorithm)

Grid size	VP ratio	Mflops	sec/step	XMP time/CM time
128×128	1	58	0.43	0.66
256×128	2	86	0.60	0.75
512×512	16	106	4.2	0.86

- Timings, Mflop rates (16k processors, explicit algorithm)

Grid size	VP ratio	Mflops	sec/step	XMP time/CM time
128×128	1	63	0.43	0.69
256×128	2	123	0.44	1.34
512×512	16	241	1.8	2.62

TIMINGS: 2D BLOCK IMPLICIT AND 3D

- 2D Timings, 16k processors, block tridiagonal algorithm

Grid size	VP ratio	sec/step	XMP time/CM time
128×128	1	2.61	0.23
256×128	2	3.21	0.37
256×256	4	4.84	0.49
512×256	8	9.66	0.49

- 3D Timings, 8k processors, explicit algorithm

Grid size	VP ratio	sec/step
$32 \times 32 \times 16$	2	1.94
$64 \times 32 \times 32$	8	3.80
$64 \times 64 \times 32$	16	6.53

PROBLEMS AND PROSPECTS

- Memory limitation
- Communication bottleneck?
- Boundary conditions
  - > Complicated boundary conditions – inefficient?
- Future plans:
  - > 3D Implicit